Information Integration

The process of integration, evolution and versioning

Ander de Keijzer Maurice van Keulen
Faculty of EEMCS, University of Twente
POBox 217, 7500 AE Enschede, The Netherlands
{a.dekeijzer,m.vankeulen}@utwente.nl

Abstract

At present, many information sources are available wherever you are. Most of the time, the information needed is spread across several of those information sources. Gathering this information is a tedious and time consuming job. Automating this process would assist the user in its task. Integration of the information sources provides a global information source with all information needed present.

All of these information sources also change over time. With each change of the information source, the schema of this source can be changed as well. The data contained in the information source, however, cannot be changed every time, due to the huge amount of data that would have to be converted in order to conform to the most recent schema.

In this report we describe the current methods to information integration, evolution and versioning. We distinguish between integration of schemas and integration of the actual data. We also show some key issues when integrating XML data sources.

Contents

1 Introduction

Nowadays, a lot of different information sources exist, ranging from large database systems used at global multinationals, to small applications on private PDA's. Large companies merge, more than ever, with other companies, or at least merge some of there departments to increase overall efficiency. These mergers have their influence on the database systems used by these companies. Either a new systems is set-up and information from all original companies has to be conformed to this new system, or one of the systems is chosen and all information from the other systems has to be transformed to fit into the new system. Either way, information is being altered, while the facts described by the information didn't change.

On a smaller scale, people at home use PDA's that can communicate with other devices. The applications used on different devices, aren't necessarily the same. They can vary in vendor and version, or a combination of both. Even when someone buys a new PDA with different software, or just upgrades the applications on an existing PDA, data should be able to be used without having to manually restore the information.

Integrating these information sources, requires a lot of work from the users of the systems involved. The structure of the data sources that need to be integrated usually differ. The data is described in various ways, or the granularity of the data stored varies.

The problems described here, are caused by evolution or versioning of the software, from one version to the next, or integration of different information sources. This report describes the current state of the art in integration, versioning and evolution of data systems.

1.1 AmbientDB

This research is carried out as part of the MultimediaN project, subproject AmbientDB. The overall goal of this project is to facilitate user friendly, global data management for the purpose of ambient intelligence. With the integration of information we will provide a global view to the data, without the user having to know where (in what information source) the data resides. The overall architecture of AmbientDB is given in Figure 1. The goal of this research is focused on the integration of information sources, indicated by the circle in the figure.

1.2 Information transformation

Integration of information sources, like databases, or data from different PDA's, requires at least one of those information sources to be transformed into a format that can be matched against one of the other information sources. This transformation is split into two separate phases. First the schemas of the data sources involved will have to be conformed. During the next step, the data integration phase, the data of the information sources is transformed to match the newly composed schema and merged into the integrated information source.

In this document we show several methods to integrate information sources. Both methods for schema integration and data integration will be discussed.

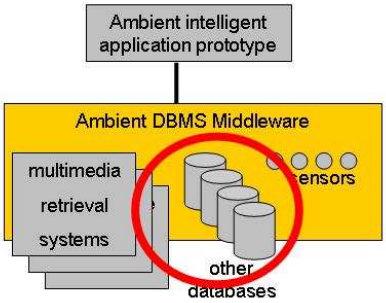


Figure 1: Overall architecture of AmbientDB

1.3 Outline

The remainder of this report is organized as follows. Section 2 gives an overview of information integration, the general steps involved and the areas in which these problems occur. Versioning and evolution of the information schema is discussed in Section 3, while data integration is discussed next, in Section 4.

2 Integration of information

This section gives a broad overview of integration of information sources, versioning and evolution. It describes what integration is, the overall process and problems of integration of information, different kinds and levels of integration, approaches to solve the problem of integration and how to query integrated data. First we define what integration of information is.

2.1 What is integration of information

Integration of information is accessing different sources of data, containing possibly overlapping information as one single source. The most common approach is demonstrated in Figure 2, others will be described later. The different sources of information may, and probably will, have different schemas to represent the data. Wrappers are used to transform the data to *fit* into the global schema. A mediator is used to access the integrated data. This mediator will translate user queries to fit the individual schemas provided by the wrappers.

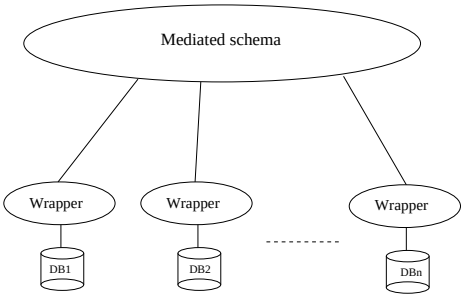
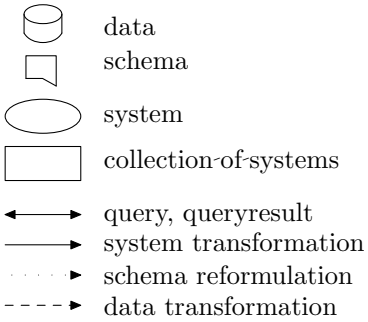
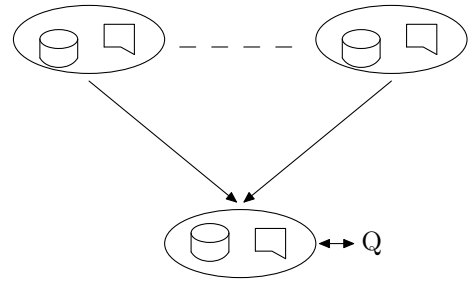


Figure 2: Integration of Information

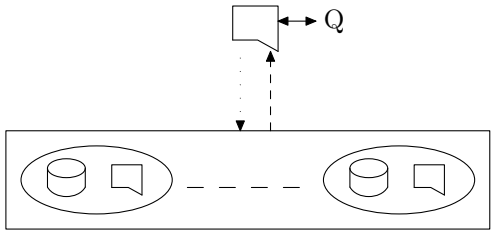
The concepts of integration, versioning and evolution, used in this report, are schematically shown in Figure 3. We use the following symbols in this figure and throughout the rest of the document



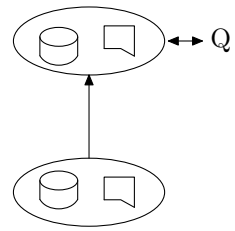
The following definitions are used:



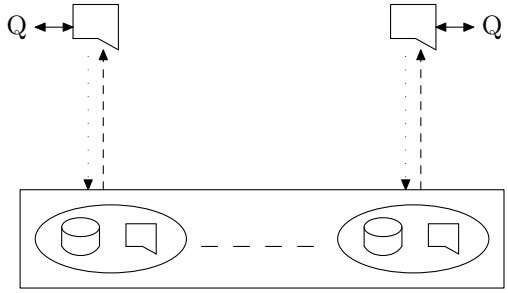
(a) Materialized Integration



(b) Mediated Integration



(c) Evolution



(d) Versioning

Figure 3: Schematic view of concepts

- Synchronization
  If one object is contained in more than one information source, after synchronization the desciptions of the object in all information sources have to be in agreement.

- Information Integration
  We distinguish between *materialized* (Figure 3(a)) and *mediated* (Figure 3(b)) information integration. In both cases, multiple information sources, containing possibly overlapping information are presented as one global information source. In case of materialized information integration, the integration process takes place at one specific time at which a new global information source is created. This new information source can than be used as a normal information source. In case of mediated information integration, the information sources are not actually integrated, but the global information source is virtual. A component, called a *mediator*, acts as the global information source. This mediator transforms queries to the global information source into queries to the underlying information sources and gathers and combines the results from these information sources, which can than be presented as the result from the global information source. The underlying information sources in this case remain autonomous.

- Schema Integration
  Information integration can be split into two phases. The first phase, integrates the schema of the information sources. The schemas of both information sources are compared and a global schema is composed.

- Data Integration
  The second phase of the integration of information is the integration of the actual data. Schemas between different sources are assumed to match. The data is transformed to match the schema of the global information source. Possible conflicts between information from both information sources are solved.

- Schema Evolution
  Schema evolution (Figure 3(c)) is accommodated when a database system facilitates the modification of the database schema without loss of existing information. When the schema of the database changes, information already contained in the database, can still be accessed, using the new schema. Also in the case of evolution, there is a distinction between materialized and a mediated. In the case of materialized, the data is converted to the new schema, whenever the schema changes. Data is always stored using the current schema. In case of mediated evolution, the data is stored using the schema used to insert the data into the database. When a query is passed to the database, the mediator transforms the schema to match the schema of the data.

- Schema Versioning
  Schema versioning (Figure 3(d)) is accommodated when a database system allows the accessing of all data, both retrospectively and prospectively, through user definable version interfaces. The data is stored in the database using the newest schema at the time the data was inserted into the database. Queries to the

database can use any of the schemas ever used in the database. A mediator will have to transform the schema to match that of the data.

The latter two definitions are taken from and further explained in [Rod95]. Evolution and versioning can be seen as specializations of information integration. In all cases, data stored using different schemas are presented as having one schema. With integration, two or more information sources are combined into one source. Evolution allows users of the data to access old data through new schemas, demanding for inteInformation integration, gration of, at least, schemas of old and new data source. Versioning allows the user to access the data through any of the old or new schemas.

There is however one difference between information integration and evolution / versioning. Schemas of information sources in a versioning or evolution setting are likely to be largely the same. The schemas all have a common *ancestor*.

At present, there is a number of existing systems that provide data integration, SimInt [LC94], LSD [DDH01], SKAT, TransScm [MZ98], DIKE, ARTEMIS [BBC+00] and CUPID [MBR01]. An overview of these systems is given in [RB01].

2.2 Why integration

The number of information sources has grown enormously. Companies and governments use several databases to store information about customers, citizens, products, regulations, etc. Many of these companies, or departments, and even (local) governments merge. As a result their information, stored in databases, needs to be merged as well.

However, also individuals have their own databases. With computer chips becoming smaller and smaller, data can be stored on tiny surfaces. PDA's today contain more data than a personal computer did several years ago and can be considered an information source. Owning a PDA is no different, and not even more expensive, than owning a paper calendar and address book. Most people also use such systems at home on their personal computer. These information sources can easily come from different vendors, or are different versions of the same vendor. However, they need to be synchronized to maintain consistency and correctness. Also synchronization, or updates between PDA's from different people should be possible, because PDA's are often used to exchange information between people.

In eBusiness, companies use different information sources for selling their products. A company may sell products from different vendors. Each of these vendors has an information source with a specific schema. Automating the ordering process, requires the schema of the selling company to be converted into schemas of the different vendors. Also, product information from one vendor can be different from product information from another vendor. The customer, however wants a one view on all products. Converting these schemas both to vendor and customer can be seen as an information integration process.

2.3 Casestudy

In this section, we will provide a motivating example on which we will base most of our next examples. Although this examle is taken from actual data[1], some parts are fiction.

Information about citizens in The Netherlands is maintained by the municipality in which the citizen lives. These municipalities have one general data source for the personal information of each of their inhabitants, this data source is called GBA[2]. Besides GBA, the municipalities have several other independent data sources, of which some are listed here:

- FIS4ALL
  Financial information.
- PIV4ALL
  Information about citizens.
- VGS4U
  Tax registration for local (municipal) taxes.
- DocMan
  Document Manager. This data source contains correspondence with citizens of the municipality.

Some data sources have overlapping information, e.g. both PIV4ALL and GBA contain information about citizens, which is an example of overlapping information. Also some data sources depend on data from other data sources, e.g. VGS4U uses FIS4ALL to register tax invoices. Although information is updated in all data sources, overlapping information should be consistent. At present most of this work is done manually, which is time consuming and very unsatisfying work. The problems are even larger when the data sources of the central government and local taxes are taken into account.

Access rights permit or prohibited certain employees at the municipal office to access some or all data sources. For example, at the person administration office, employees are allowed to acces and alter PIV4ALL and GBA, but not the other data sources. At the tax office, access to FIS4ALL and VGS4U is granted. There are many different offices and employees with different kinds of access rights. As a result all possible combinations of access to the available data sources are present within one municipality.

There is a need to automate these processes. Not only because of the work it would save, but also to ensure correctness and consistency.

2.4 Kinds of integration

There are two kinds of integration of information sources. The first one, integrates the data at a given point and then materializes this integrated source. From this point on,

[1] The example was provided by eMaxx, one of the participants in the AmbientDB subproject of MultimediaN.

[2] which stands for Gemeentelijke Basis Administratie (Municipal Basic Administration).

the information can be accessed through the newly created information source. We will refer to this kind of integration as *materialized integration*.

The second kind of integration is achieved by designing a mediated schema on top of existing information sources. These information sources may need a wrapper to through which can be communicated with the mediated schema. This kind of integration is shown in Figure 2. We will refer to this kind of integration as *mediated integration*.

The advantage of mediated integration is that the original information sources still exist after integration. If, for some reason, an application still wants to use this information source in its original form this is possible. Especially when information sources are the property of different companies, this is a big advantage. Mediated integration also ensures that if the integration process introduces errors in the information, the original information isn't lost. Naturally, data obtained with this method is still inaccurate, but the correct data could be retrieved manually. The advantages of materialized integration over mediated integration are that the integration process occurs only once. Queries to the integrated data source will therefore be quicker with the materialized integration than with the mediated integration. Since this integration step only occurs once, it doesn't matter if it takes longer. More advanced methods can then be used to integrate, ensuring a better result in the integrated data.

2.5 Levels of integration

Integration of Information sources has to be performed at two levels. First the schemas of the sources has to be integrated. Mapping rules from a source to the resulting information source have to be designed. When these mapping rules are known, data contained in the information sources has to be integrated into the new information source.

Information integration can be seen as a combination of both schema integration, followed by data integration. The data integration step, however, depends on the result of the schema integration step and cannot be performed independently.

Consider two address books as given in Table 1. If we merge these address books, first the schemas of both books have to be transformed to match. To the user it is apparent that *room* in address book A matches *location* in address book B and also that *name* in book A matches the concatenation of *firstname* and *surname* of book B. However, *phone* in address book B cannot be matched with an attribute in address book B. In other words, the schema (name, room) has to be matched with schema (surname, firstname, location, phone).

If the schemas are integrated, the data from both original information sources has to be transformed to match the resulting schema. Next, the data from the different information sources has to be integrated, which means that duplicates have to be eliminated, conflicts have to be solved and possibly missing information has to be dealt with. The data itself may also have to be converted to match the new schema. This process is illustrated in Figure 4.

Techniques and methods used to integrate information sources at schema level is discussed in detail in Section 3. Different approaches to integration at the data level is described in Section 4.

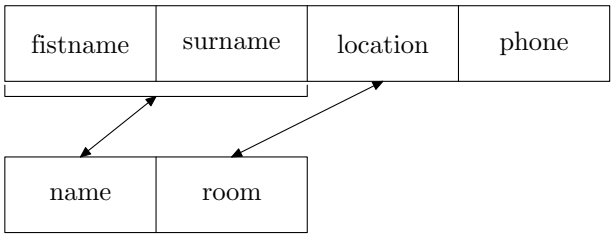


Figure 4: Matching of Schemas

name	room
John Doe	3122
Ed King	2012

(a) Address book A

surname	firstname	location	phone
Doe	John	3122	4243
King	Ed	2012	3519
Hamburg	Mark	2022	3530

(b) Address book B

Table 1: Two address books

2.6 Differences in data models

Integration of information sources can be based on a number of properties of the original information sources, e.g. attribute names or attribute values. The data model used for the information source has a great influence on which property can be used. Integration of object oriented databases as in [Thi95, Ver97] is based on structure and behaviour of the information source. Structure of an object oriented database is represented by its attributes, while behaviour in these databases is represented by the methods.

Evolution, and therefore also versioning, of object oriented databases is especially difficult, since both structure and behaviour can change from version to version. Add [Zic92]

XML information sources, however, are not capable of representing behaviour, but structure is very suitable to be used as an integration property in XML. Relational databases have attribute names and (foreign) keys on which integration can be based.

2.7 Data retrieval

Querying an integrated information source should be no different than querying an ordinary information source. With a materialized integrated information source, this is automatically true, since the result of the integration process is a new information source. This information source is the global view of all information sources used in the integration process.

For materialized integration, evolution and versioning, however, queries to the global information source have to be changed. First the mediator has to determine which information source contains which part of the requested information. Next, queries to the individual information sources have to be formulated. Results from these information sources are combined and transformed to match the original schema of the query. This information is then returned as the result to the query.

Retrieving and querying data from these integrated sources is discussed in detail in chapter 5.

2.8 Semantics

Although, the integration process should be automated as much as possible, the user will at some point, be involved. If the system is able to fully integrate information sources without immediate help from the user, errors in the integrated information source are possible and even likely to happen. The main reason for this errors is the semantics of the information source.

For the user, these semantics are mostly very easy to understand. For a device, the semantics is much harder to obtain. For example, if we have two information sources. Source A contains information about employees at the university and has schema (name, room, number), where name is the name of the employee, room is the room number and number is the phone number. Source B contains information about students of the same university. The schema of source B is (name, address, number), where name is the name of the student, address is the home address and number is the student id number.

For humans the integration of these two information sources would be relatively easy. Both employees and students have an address, which might be just a room number. Employees have a phone number, but students do not and students have an id number. Automating this process would probably result in phone numbers from employees and id numbers from students being merged. This, clearly, is an undesired effect.

3 Reconciling schemas

The integration of information starts at the schema level. As shown earlier, schemas of the original information sources have to be transformed in order for the data to be integrated. The transformation of these schemas is preceeded by a matching of schemas, i.e. what parts of schema 1 match with what parts of schema 2. The methods used to match these schemas depends on the kind of schemas used in the information sources. Also, if instead of integration of data sources, evolution or versioning is used, matching will most likely be easier, since both schemas originate from the same ancestor schema.

The process of schema integration is discussed in section 3.1. Wrappers are an important aspect of integration and these will be the topic of section 3.2. In section 3.3 we will discuss the role of mediators and the way in which they can be designed. In section 3.4, matching techniques are discussed, taking into account the different types of schemas we can encounter. Using the notion of time is helpful with evolution and versioning, this will be the topic of section 3.6.

3.1 Integration process

The process of integrating different schemas into one global schema [SL90], can be divided into four different phases [BLN86]:

- Preintegration
  During this phase, the integration method is chosen and additional information is gathered.

- Comparison
  Source schemas are compared and similarities and differences are detected

- Conformation
  If conflicts in the schemas are detected, an attempt is made to resolve these conflicts.

- Merging and restructuring
  The results of the conformation phase are merged and restructured to ensure minimality and understandability.

3.2 Wrappers

Wrappers provide a common interface to a data source. If we have a data source A and a data source B, that both have different schemas, then wrappers translate the interaction from the user, specified in using the global schema, to the schema of the data source. Consider the two data sources shown in Figure 2. The schema which the user uses to pose queries could be $(name, room)$. In this case, firstname and name from data source A should be mapped to name in the global schema and building and room in data source B should be mapped to room. But wrappers should also be able to map the other way, i.e. from user schema to actual data source.

name	firstname	room
Doe	John	ZI-3122
King	Ed	ZI-2012

(a) Data source A

name	building	room	phone
John Doe	ZI	3122	4243
Ed King	ZI	2012	3519

(b) Data source B

Table 2: Two Semantically similar data sources with different schemas

Besides converting between schemas, a wrapper is also used to convert between data models. If the global information source is presented as an XML source, underlying information sources can still be relational, object-oriented, or any other kind of information source. The function of the wrapper is to convert these relational and other kinds of information to equivalent data in the XML schema.

In short, the function of a wrapper is to provide a common interface to the underlying data source. The mediated schema in Figure 2 can access all data sources through a common interface provided by the wrappers in between the mediator and the actual data source.

Designing these wrappers for various data sources is a tedious and time consuming process. In the TSIMMIS project [CGMH$^+$94, GMPQ$^+$97] wrappers are created semi-automatically.

3.3 Mediators

A mediator acts as integrated information source for several independent autonomous information sources (see Figure 2). Queries passed to the mediator are translated into queries to underlying information sources and the resulting information is combined and presented as result of the mediated information source. The underlying information sources are usually accessed through wrappers.

The TSIMMIS system [GMPQ$^+$97, GMHI$^+$95] is a mediated system. The system uses the Object Exchange Model (OEM) to query to data.

Mediating can be looked at from different perspectives. The first, and obvious, way to mediate is Global as View, or GAV. With GAV the mediated schema is composed of the different original information sources. The second way to mediate is using Local as View, or LAV. Here the different autonomous information sources are considered to be subsets of the mediated schema. We will show both GAV and LAV in the following sections.

3.3.1 Object Exchange Model

The mediator distributes the query over the different available information sources. The mediator can do this using a uniform schema kind. The main focus of the mediator in case of a query is distributing the query over the information sources itself, collecting the different parts of the information from all information sources.

In the TSIMMIS system this uniform schema is the Object Exchange Model (OEM) [GMPQ+97, PGMW95]. Data in OEM is *self-describing*, i.e. it contains its own schema and does not need an external schema. Objects in the model consist of four parts. They have an *object ID*, a *label*, which describes what the object represents. Labels are human readable and therefore contain all information needed about the object. The third part is a *type* of the value. The type can be either atomic or *set*. The last part of an object is the *value*, which is either an atomic value, or a set of objects.

Our address book example in OEM would look like

```
<set-of-addresses, set, {ad_1, ad_2}>
  ad_1: <name-and-address, set, {name_1, room_1}>
  ad_2: <name-and-address, set, {name_2, room_2}>
    name_1: <name, string, 'John Doe'>
    room_1: <room, string, '3122'>
    name_2: <name, string, 'Ed King'>
    room_2: <room, string, '2012'>
```

Wrappers will transform the query distributed by the mediator to the local information source. This might entail rewriting the query to a different language that can be handled by the information source, or using a different schema kind.

3.3.2 Global as view

With Global as View, the mediated information source is viewed as a composition of the original information sources. Suppose we have three information sources, addressbook (name, room, phone), addresses (name, room) and phones (name, phone). In all sources attribute *name* is key. We want to create a mediated information source which provides the same information as as initial address book example. We can compose a mediated view completebook as follows

```
CREATE View completebook AS
  SELECT * FROM addressbook

  union

  SELECT name, room, phone
  FROM addresses, phones
  WHERE addresses.name=phones.name
```

In this example the mediated view completebook is composed of the union of addressbook with the join of addresses and phones.

A drawback of Global as View is that extending the system with additional information sources is hard, since the mediated schema has to be redesigned and rebuilt.

3.3.3 Local as view

Local as View considers the original information source to be derived from the mediated schema. It describes the information sources in terms of queries to this mediated

information source. Given the previous completebook example, three views would be created, for addressbook, addresses and phones.

```
CREATE SOURCE addressbook AS
  SELECT * FROM completebook

CREATE SOURCE addresses AS
  SELECT name, room FROM completebook
```

and

```
CREATE SOURCE phones AS
  SELECT name, phone FROM completebook
```

The *information source* completebook does not exist, but is a virtual information source composed of actual information sources addressbook, addresses and phones. Adding an information source in this case, is easy as the information sources are independent of each other. Adding one source would result in adding one extra query.

3.4 Matching

Before schemas can be integrated, they have to be matched first. A formalization of the schema matching problem is given in [SvKJ05]. In [RB01] an overview is given of approaches to automatic schema matching. A hierarchical classification of schema matching approaches is presented. Schemas can be matched based on a number of characteristics. Names of attributes or elements can be compared. If they match, or are relatively similar, the attributes are matched and the associated data can be integrated. Instance data can also be compared. If the data contained in an attribute is similar, it is likely that the attributes have to be merged. These methods base their decision on one property of one attribute and are therefore referred to as *element matching*. Other approaches use the structure of the schema itself, or a combination of attributes. These methods are called *structure matching*.

When trying to match two schemas R_1 and R_2, a number of situations can occur [BLN86]. The schemas can be

1. *Identical*. In this case R_1 and R_2 are exactly the same. This is the case for systems for which the same modeling constructs are used and the same perceptions are applied.

2. *Equivalent.* In this case R_1 and R_2 are not identical. This is caused by use of different modeling constructs. However, the perceptions applied are the same and must be coherent. Definitions of equivalence are usually based on three different types:

   (a) *Behavioural*. Two schemas are equivalent if for every instance of one schema, there is a corresponding instance of the second schema, that gives the same answers to every possible query.

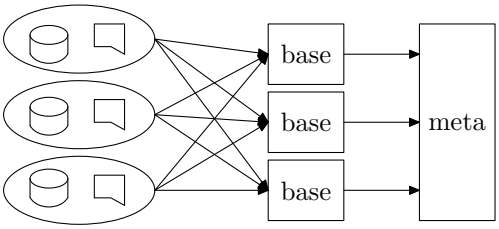


Figure 5: Using Learners

(b) *Mapping*. This is the case, when instances of both schemas can be put in a one to one correspondence, see [Ris77].

(c) *Transformational*. This types of equivalence holds, if one of the schemas can be transformed into the other by using atomic transformations that make the schemas behavioural or mapping equivalent.

3. *Compatible*. The two schemas are neither identical, nor equivalent, but they are also not in contradiction. This means that modeling constructs, designer perception and integrity constraints cannot be contradictory.

4. *Incompatible*. If two schemas are not identical, equivalent or compatible, they are incompatible. The two schemas contradict, because of the incoherence of the specification.

3.5 Learners

In the Learning Source Description (LSD) system [DDH03], learning modules, called *base learners*, are used. Each base learner uses well one certain type of information to find semantical mappings between schemas.

There are many different base learners. Examples include: Name Learners, Naive Bayes Learner, Content Learner and XML Learner. The Name Learner matches elements based on their name. In case of XML, this name is the tag name. The learner can use lists of synonyms to match similar, but not equal names. The Content Learner matches elements based on the content of the attribute. In case of XML, this is the data value of the XML element. The learners use a set of recognizers, which are each capable of recognizing a certain kind of item, i.e. a zip code or a phone number. A schematic example of learner process is shown in Figure 5.

The base learners are first trained in a phase called the *training phase*. During this phase, the user has to confirm the results so the learner *knows* if the results are correct. After this training phase, the learner can be used, this is called the *matching phase* [Hal04].

Base learners now how to map on schemas, using one technique. Meta Learners combine the results of base learners to match using multiple techniques. See [DDH03].

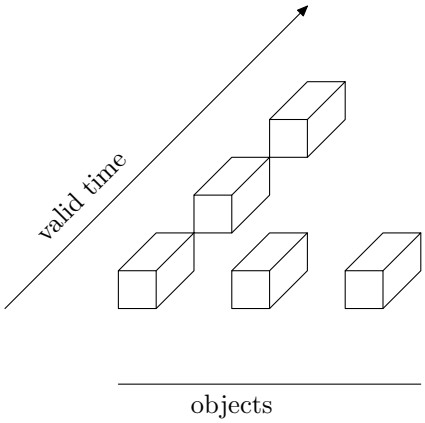


Figure 6: Valid Time

3.6 Using time

If we allow the schema of an information source to change over time, data stored at different moments can have a different schema. A query to this information source should, at a minimum (evolution), be possible using the latest version of the schema. To keep track of the version used for particular parts of the data, a notion of time can be added to the schema and data. There is much overlap with this kind of time usage and time in temporal databases.

Different notions of time can be used, to accommodate for different functionality of the system. Keeping track of the original schema of a particular part of the data is one aspect, but also rolling back the information source to a prior state, or being able to answer time related queries, are possibilities of using time.

3.6.1 Valid time

The first notion of time that can be included in an information source is the time at which the recorded event occurred. Suppose we change the address of one of the persons in our address book, indicating this person has moved. We can attach a time stamp to this record indicating from which point in time this record is valid. This kind of time is referred to as *valid time*.

With valid time, we can ask the information source questions like: *What was the address of person X on February* 2^{nd} *last year*. A schematic representation of valid time is given in Figure 6

3.6.2 Transaction time

Another notion of time, is the time when the record was inserted into the information source. If all the records are associated with such a time stamp and provided that deletions are also recorded, we can reconstruct the information source for a certain date

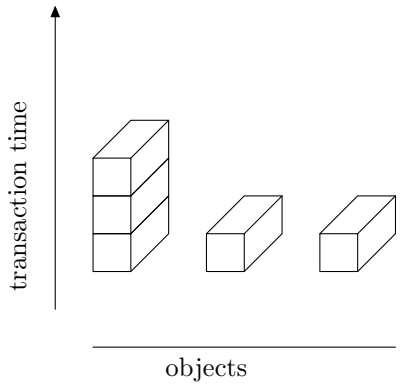


Figure 7: Transaction Time

and time. The time recorded is the time at which the transaction occurred that created that particular record, and is therefore referred to as *transaction time*. Transaction time enables the system to do a roll back to any particular point in time. This is shown in Figure 7.

3.6.3 User defined time

The last notion of time we discuss, is already present in current relational database systems. If the user wants to represent time, he can use the DATETIME type. The database system itself regards this information as just another type of data. From the database point of view there is no special meaning associated with this type of time and it is therefore known as *user defined time*.

3.7 Semantics of schema

As indicated earlier, the semantics of schemas is important when integrating data sources. Only semantically similar data can meaningfully be integrated. A possible solution to this problem, and also commonly used, is to let the user provide the semantics. Most of the time, this is done by letting the user specify how the data has to be integrated. In the case of the LSD system, this is achieved by first specifying mappings manually in the training phase. Next, in the matching phase, the system can match the schemas automatically.

Another method to collect semantical information is to use an ontology, indicated by [Ver97]. This process, however, is error prone and calls for an accurate and up to date ontology. Another problem that arises when using ontologies for integration of information sources, is that they would have to support versioning and/or evolution as well [NK04].

3.8 Distributed integration

Integration of information can also be accomplished in a distributed setting. Different clients *own* different parts of the overall information source. In this case, not all clients have to be available at a particular time, therefore the mediated schema has to adapt to the information sources available. A distributed setting is shown in Figure 8

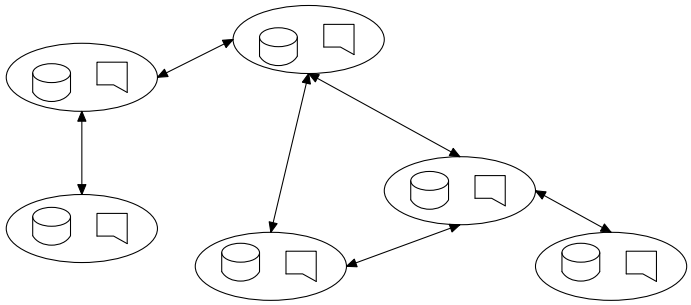


Figure 8: Distributed environment

4 Data integration

After the schemas of two data sources have been integrated, the data contained in these information sources has to be integrated as well. This chapter describes the problems arising when data is conformed to the global schema. In section 4.1 the requirements for data integration are discussed. Section 4.2 shows several problems when integrating data into the global schema, using the mappings from the schema integration phase. In section 4.3 we will show one novel technique for conflict handling.

Many of the methods mentioned in the context of reconciling schemas can also be used in data integration. Especially, learners are a good method to improve data integration. Using feedback from the user can improve the integration result. We will, however, not mention these methods here again.

4.1 Requirements

The newly integrated information should contain the same information as the *union* of the original information sources. This means that the integration process has to preserve the data contained in the information sources, i.e. if an object is contained in one of the original information sources, it should be contained in the resulting information source. This requirement is referred to as *preservation*.

Another requirement is that of *consistency*. The appearance of an object should not have been changed by the integration process. For example, if we integrate two address books and one of these books contains persoon John with room number 3122, then the resulting information source should contain John (preservation) and his phone number should be 3122 (consistency).

4.2 Matching and merging

The result of the schema integration phase is a set of mappings from a source schema to the global schema. There are four kinds of mappings possible, i.e. one to one, many to one, one to many and many to many.

One to one mappings are easiest to integrate. In the simplest case, the attribute value just has to be copied the the global information source. We will, however, show that this is not always the case. An example of a one to one mapping is shown in Table 1. If these two tables are integrated, attribute room maps to attribute location and attribute values from one table can just be copied to the other table.

With many to one mappings the integration process becomes more difficult. If we map table B from Table 1 onto table A, there is a many to one mapping from (firstname, surname) to name. The rule that has to be specified for this conversion is

```
A.name = B.firstname + ' ' + B.surname
```

If we compare this rule to the schema integration rule (B.firstname, B.surname) $\mapsto$ (A.name), we see that the data integration rule is much more complicated, because it specifies *how* the data is transformed, whereas the schema integration rules just specify *which* attributes are mapped.

One to many mappings are even harder than the many to one mappings. Consider the previous example, but with a mapping from A to B. The schema mapping rule then becomes (A.name) $\mapsto$ (B.firstname, B.surname), but specifying the data integration rule is not as easy as before. If we take 'John Doe' as an example, we can just split the name at the space, obtaining John as a first name and Doe as the surname. This rule could be specified as

```
(B.firstname, B.surname) = split(A.name, ' ')
```

where split is a function that splits the first parameter at the first occurrence of the second parameter. If we use this specification and take the typical Dutch name 'Jan de Boer', the result ('Jan', 'de Boer') is not perfect, since sorting on the surname should normally result in 'Jan the Boer' being sorted on 'Boer' instead of 'de Boer'. The result, however, is acceptable, but if we use a more complicated Dutch name like 'Jan Hendrik de Jong', the specification fails. The firstname of this person consists of both Jan and Hendrik, but the specification will only use Jan.

For obvious reasons, the many to many mappings are more diffecult than one to many and many to one. As an example, we will use the name mappings, but with an additional attribute called prefix. The word 'de' in the lastname of 'Jan de Boer' is called a prefix and often stored separately. If we have two information sources, A with schema (firstname, address) and B with schema (firstname, prefix, surname, location), mapping the name from one schema to the other is a many to many mapping.

4.3 Uncertain and imprecise data

Instead of matching the data and storing just the matches, while discarding all *non matches*, match results can be assigned a *level of confidence*. This level of confidence is presented by a number, ranging from 0 to 1, where 0 indicates a total mismatch and 1 indicates a perfect match. If the matches and their confidence levels are stored in the resulting information source, more data will be stored, but there will be no data loss. This method, however, will result in a ever growing information source, and very soon the information will be to large to handle. A simple, but effective solution is only to store attributes with a confidence level larger than a certain α.

One model describing uncertain data is the possibilistic relational model [BDP03]. An attribute value in this model may be a possibility distribution, indicating for each possible attribute value the level of confidence. Table 3 shows a possibilistic address book, where the room of John Doe is uncertain. The level of confidence for room 3035 is 0.8, while the level of confidence for room 3122 is 0.5.

name	room
John Doe	0.8/3035 + 0.5/3122

Table 3: Possibilistic address book

Another possibility to handle uncertain data is by using cisets [Nai03]. With cisets, both the level of confidence and the level of disprove are stored. This methods allows

for a more accurate registration of certainty. If there is a lot of *evidence* a certain attribute value is false, than the level of disprove is increased. If, at the same time, a lot of evidence is collected that the same attribute value is true, the level of confidence is increased as well. In this case, there is a lot of doubt about the attribute value.

A somewhat simpler way to handle uncertain data, is the probabilistic data model. It can be compared to the ciset solution, where the level of confidence + the level of disprove equals 1. Probabilistic databases are explained in section 6.2.

5 Querying

This section will show some methods to query data that results from one of the previously mentioned processes, such as mediated data, probabilistic data or time enriched data.

5.1 Object Exchange Model

The Object Exchange Model, described in section 3.3.1 allows a mediator to communicate with all wrappers using a single data model. Besides the data model, a query language is designed. This query language, called Mediator Specification Language (MSL) [GMPQ+97] allows declarative queries to be formulated. A query constists of a head, followed by :- and a body. The head describes objects that are requested, compare with the SELECT statement in SQL. The body describes the conditions that have to be met, compare with the WHERE statement.

The following query retrieves the room from person 'John Doe' in addressbook A in Table 1

```
<room X> :-
  <addressbook { <person {<name "John Doe"> <room X>}> }>@A
```

Here the room number is returned from the person with name 'John Doe', who is a person in the addressbook, using information source A.

5.2 Uncertain data

Uncertainty in the data entails that query results will contain uncertainty as well. Consider the address book example from Figure 3. If we were to ask for the room of John Doe, the database contains both rooms 3122 and 3035 as a result.

A different approach to querying uncertain data, is that of *possible worlds*. Using the possible world approach when querying uncertain (relational) data provides clear and sound results. In our example, both rooms have an associated probability, creating possible worlds with 2 variants for real world object John Doe. The result to the query will have to reflect the existence of these two possibilities.

Using aggregates in the query normally summarize data. If we view a normal database a description of the real world and a probabilistic database as a description of multiple possible worlds, then aggregates can be applied accordingly. Probabilistic aggregates summarize the data in all possible worlds and associate the probabilit of that particular possible world with the summary. Querying of probabilistic relational data with the possible world approach is described in [dKvK04]. Querying possibilistic XML using the possible world approach is described in [vKdKA05]

There are other ways to query probabilistic data, that are not related to the possible world approach. The semantics behind these methods is often harder for humans to grasp, but is aimed more at computers. Querying probabilistic data is described in [dKvK04, vKdKA05].

Possibilistic databases are queried in a similar manner. Also in this case the result will contain uncertainty [BP04]. More specifically, the result will be a possibilistic relation.

Independent of the the way in which data is presented or queried, the wrappers used will have to be aware of the fact that uncertain information is present in the information source, because the result will also contain uncertain information. Depending on the application, the wrapper will have to either return the uncertain data in some form, or convert the data to a certain representation.

5.3 Querying with time

In [MS87] the algebra is extended with transaction time. This means that when querying the data, the time that that the data was entered can be specified. The transaction time allows users to roll back the system, which makes it possible to create snapshots of the data of any desired time.

6 XML specific issues

Most of the methods presented in this report can be applied to XML integration, versioning and evolution. However, some methods are diffecult to implement in case of XML, due to its properties. Also these properties sometimes allow for different approaches or provide additional information that can be used to simplify integration, versioning or evolution.

6.1 Versioning

In order to support versioning, instead of the data itself, changes to the XML document can be stored [CTZ01]. Possible changes are *insert* and *delete*. An update of the data can be accomplished by first deleting the object and then inserting the new value.

In [CTZ01] a method is described to obtain any previously stored version of the document. Since an XML document contains both data and structure, any version of the schema can be obtained as well.

6.2 Probabilistic data

When data is integrated, one of the problems that may arise, is that of data loss. Two objects from different sources that seem similar are combined in the final data source. When in fact, these merged descriptions are not representing the same object, information about at least one of the objects is lost. This problem could be solved, if we saved both instances, however, if we just copy them both in the resulting data source, the purpose of integrating the two sources is lost. With probabilistic databases, we have the means to indicate that both descriptions *could* be about the same real world object. For one particular real world object, we can also indicate that there are multiple possible appearances, of which one is more likely than the other.

6.2.1 Model

There are several models introduced to hold probabilistic data. All models are based on the same principle, data is assigned a probability. In the case of relational data, probabilities can be either assigned to a relation itself, or to attribute-values contained in the relation. The former type of probabilities is said to be of *Type-1*, while the latter is of *Type-2*. Both types of probabilities are shown in Table 4.

	name	room
[0.2]	John Doe	3035
[0.8]	John Doe	3122

(a) Type-1 probabilities

name	room	
John Doe	3035	[0.2]
	3122	[0.8]

(b) Type-2 probabilities

Table 4: Two types of probabilities

Both relations in this example assign a probability of 20% to the fact that John Doe occupies room 3035. However, in case of the Type-1 relation there could exist two persons, both with name John Doe, but occupying a different room, whereas in the Type-2 instance, there can only be one person John Doe, who either occupies room 3035, or room 3122. This is due to the fact that in case of a Type-1 probability, the probability is associated with the relation itself. In our example, there is a 20% chance that the relation (John Doe, 3035) exists. A Type-2 probability is associated with an attribute-value, therefore the relation itself remains if the attribute-value itself doesn't exist. In other words, there is a 80% chance room 3035 is not occupied by John Doe (according to the Type-2 example), but this has no effect on the person.

A side effect from the differences in the two types of probabilities, is the fact that with Type-1 probabilities, there can be 0, 1 or 2 persons in the database. There is a chance of $(1 - 0.2) \times (1 - 0.8) = 0.16$ that there is no person in the database. There is also a chance of $0.2 \times (1 - 0.8) + (1 - 0.2) \times 0.8 = 0.68$ that there is one person in the database and a remaining chance of $0.2 \times 0.8 = 0.16$ that there are two persons in the database. The Type-2 probability ensures the existence of just one person. The object must exist, because the chance of the relation is 1, but it is impossible that there is more than one person.

These two types of probabilities can also be combined. This enables a data integrator to model the situation where it is likely that two persons in different data sources could be different persons, but could also be the same person, in which case the appearance is uncertain.

Models of probabilistic data, both relational and XML are described in [BGMP90, ZP97, DS96, dKvK04, vKdKA05, LLRS97, NJ02, ELW01, Nai03].

6.2.2 Semantics

Proper use of probabilistic data calls for a sound semantic underpinning. If the semantics of the data is clear, arguing about uncertain data is easier. Furthermore, the semantics has to be suitable for use in a data integration application. A sound and natural semantic background is that of *possible worlds*. Each possible appearance of a real world object is called a *possibility* en a combination of one possibility for every existing real world object is called a *possible world*. The notion of possible worlds is discussed in [ZP97, dKvK04, vKdKA05].

6.3 Issues

The first thing that can be observed about XML, is that due to the semi-structured way, integrating the schema is difficult. The schema is contained within the data source itself and without a DTD or XMLSchema, the schema is unknown to the system and can change from object to object. Even worse, semantically different kinds of objects can be contained in one source document.

The same problem we saw with XML schema integration, the lack of knowledge about the schema, is a problem for data integration. Unless the XMLSchema of the document is known, types of elements are unknown. Attribute types can normally be used, as a base learner, to estimate or improve matching. In the case of XML data

integration, this information is unavailable. Also, in contrast with object oriented data, behaviour can not be inferred from the data.

7 Future research

In this report we have shown the process of integration, evolution and versioning of information sources. Many of the steps involved in these processes are still open issues, or have just been partially solved, with solutions only for specific domains.

Matching algorithms at present are specialized for certain domains. Most algorithms also only handle one to one mappings. Designing and implementing many to one, one to many and many to many, more general matching algorithms is one of the main reaserch areas. Such algorithms would greatly improve integration of information sources.

The term *understandability*, used in the requirements of reconciling schemas has to be formalized. At present, understandability is subjective and therefore different for every person using the system.

Solving the problem of semantically different objects in the same XML document. Designing a detection mechanism for different kinds of objects, so they won't be matched even if attribute names and/or values are similar.

References

[BBC+00] Domenico Beneventano, Sonia Bergamaschi, Silvana Castano, Alberto Corni, R. Guidetti, G. Malvezzi, Michele Melchiori, and Maurizio Vincini. Information integration: The momis project demonstration. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 611–614. Morgan Kaufmann, 2000.

[BDP03] Patrick Bosc, Laurence Duval, and Olivier Pivert. An initial approach to the evaluation of possibilistic queries addressed to possibilistic databases. In *Fuzzy Sets and Systems*, 2003.

[BGMP90] Daniel Barbará, Hector Garcia-Molina, and Daryl Porter. A probabilistic relational data model. In François Bancilhon, Costantino Thanos, and Dennis Tsichritzis, editors, *Advances in Database Technology - EDBT'90. International Conference on Extending Database Technology, Venice, Italy, March 26-30, 1990, Proceedings*, volume 416 of *Lecture Notes in Computer Science*, pages 60–74. Springer, 1990.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[BP04] Patrick Bosc and Olivier Pivert. Possibilistic databases and generalized yes/no queries. In *Supporting Imprecision and Uncertainty in Flexible Databases*, 2004.

[CGMH+94] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[CTZ01] Shu Yao Chien, Vassilis J. Tsotras, and Carlo Zaniolo. Xml document versioning. *SIGMOD Rec.*, 30(3):46–53, 2001.

[DDH01] A.H. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. ACM SIGMOD Conf.*, pages 509–520, 2001.

[DDH03] A. Doan, P. Domingos, and A. Halevy. Learning to match the schemas of data sources: A multistrategy approach, 2003.

[dKvK04] Ander de Keijzer and Maurice van Keulen. A possible world approach to uncertain relational data. In *Supporting Imprecision and Uncertainty in Flexible Databases*, 2004.

[DS96] Debabrata Dey and Sumit Sarkar. A probabilistic relational model and algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[ELW01] Thomas Eiter, Thomas Lukasiewicz, and Michael Walter. A data model and algebra for probabilistic complex values. *Annals of Mathematics and Artificial Intelligence*, 33(2-4):205–252, 2001.

[GMHI+95] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. rey, and U. Jennifer. Integrating and accessing heterogeneous information sources in tsimmis, 1995.

[GMPQ+97] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[Hal04] Alon Halevy. XML and data integration. In *EDBT Summerschool presentation*, 2004.

[LC94] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 1–12. Morgan Kaufmann, 1994.

[LLRS97] Laks V. S. Lakshmanan, Nicola Leone, Robert Ross, and V. S. Subrahmanian. ProbView: a flexible probabilistic database system. *ACM Transactions on Database Systems*, 22(3):419–469, 1997.

[MBR01] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proc. 27th Int.Conf.on Very Large Databases*, pages 49–58, 2001.

[MS87] L. Edwin McKenzie and Richard T. Snodgrass. Extending the relational algebra to support transaction time. In Umeshwar Dayal and Irving L. Traiger, editors, *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, California, May 27-29, 1987*, pages 467–478. ACM Press, 1987.

[MZ98] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 122–133. Morgan Kaufmann, 1998.

[Nai03] Premchand S. Nair. *Uncertainty in Multi-Source Databases*. Springer Verlag, ISBN 3-540-03242-8, 2003.

[NJ02] Andrew Nierman and H. V. Jagadish. ProTDB: Probabilistic data in XML. In *Proceedings of the 28th VLDB Conference*, 2002.

[NK04] Natalya Fridman Noy and Michel C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, 2004.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, December 2001.

[Ris77] Jorma Rissanen. Independent components of relations. *ACM Trans. Database Syst.*, 2(4):317–325, 1977.

[Rod95] J. Roddick. A survey of schema versioning issues for database systems. In *Information and Software Technology*, pages 383–393, 1995.

[SL90] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 1990.

[SvKJ05] M. Smiljanic, M. van Keulen, and W. Jonker. Formalizing the XML Schema Matching Problem as a Constraint Optimization Problem. In K.V. Andersen, J.K. Debenham, and R. Wagner, editors, *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA), 22-26 August 2005, Copenhagen, Denmark*, volume 3588 of *LNCS*, pages 333–342. Springer, August 2005.

[Thi95] C. Thieme. *Schema integration based on structure and behaviour*. PhD thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, 1995.

[Ver97] M.W.W. Vermeer. *Semantic Interoperability For Legacy Databases*. PhD thesis, University of Twente, October 1997.

[vKdKA05] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic XML approach to data integration. In *Proceedings of the International Conference on Data Engineering, ICDE 2005*, 2005.

[Zic92] Roberto Zicari. A framework for schema updates. *Building an Object-Oriented Database System: The story of O_2, François Bancilhon and Claude Delobel and Paris Kanellakis Eds.*, 1992.

[ZP97] E Zimanyi and A Pirotte. Imperfect information in relational databases. *Uncertainty Management in Information Systems, A. Motro and P. Smets, Eds.*, 1997.